Vol. 3 No. #33 April 1994

LOADSTAR® LETTER

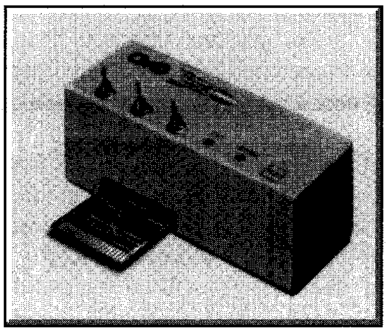
Taking It to Another Level

CMD Prepares to Ship SuperCPUs

LOADSTAR Beta Testing SuperCPU

CMD is in the final stages of testing and tweaking the 20MHZ SuperCPU accelerators. LOADSTAR is lucky enough to be on the short list for beta testing. Check us out next month for details on compatibility and performance.

We'll test the accelerators with LOADSTAR programs and commercial applications and let you know what's great and not so great. Considering CMD's amazing success with RAMLink compatibility, we expect smooth sailing. These guys really know what they're doing!


CMD's proposed SuperCPU

Jeff Fesses Up to Another Bug

By Jeff Jones. Almost from the beginning a few people had major problems with *Starbase 419* by Gus Vakalis. It seemed that anyone who actually beat level two could never coax the program to move to level three.

Well, after a small investigation followed by an attempted cover up, I, Jeff Jones, accept responsibility for adding the bug to Gus' program. It all started when I compressed his forty-level game using a BASIC pre-incarnation of *RL-EASE*, published on the same issue. When you have a forty-level game and each level takes up four blocks, you start thinking compression.

We'll be republishing *Starbase 419* in archived form. Inside the archive will be the program as supplied by the author. My apologies to Gus for contaminating his program.

Menu Toolbox II Adds High Level Power

By Jeff Jones. On LOADSTAR #144 *Menu Toolbox* will return with BASIC, compiler and machine language level access to *all* commands. So now any programmer can look to the toolbox for prewritten, pretested workhorse routines.

To make programming even less of a hassle, an instant screen command is included. With one SYS, you can set up a screen, background and menu.

LSU-S Creature

By Jeff Jones. It was a biped, intelligent, and enrolled at LSU in Shreveport. It owned a computer -- a Pentium-based machine with a 1.2 gigabyte hard drive. As I spoke with the

CONTENTS

creature, it informed me that it owned Photoshop Deluxe, a $500+ program which I purchased legally. It bragged that it was connected with pirates on the Internet and could get hold of anything -- *anything*! Just let it know. It was very excitable.

It then informed me that its computer was currently unusable because it had illegally gathered so much software that his PC had been busy all day crunching its modem's booty. It was hoping that it had room left on its hard drive to actually *use* its computer for some homework.

Gads, I've known pirates, but never one that seemed addicted to a life of crime. If my late-night classroom is transformed into Ruby Ridge by the software police, I'll let you know.

Accelerator Timing Loops

By Jeff Jones. *"If the sound of a tree falling in your program lasts for only a 40th of a second, would anyone hear it?"* The answer is probably no. Well if you've designed a sound to last for a half second on an unaccelerated C-64/128, whether compiled or BASIC, it probably *will* last only a 40th of a second accelerated.

Einstein said it best, *"The rate at which sound moves through time is regulated by crude FOR NEXT loops."* Okay, maybe Einstein never said that, but he understood relativity, and time is very relative in a FOR NEXT loop. Compile or accelerate that FOR NEXT loop and it's suddenly taking much less time to execute. And when your sounds are perfectly timed to exist for a certain number of moments, they become different or even inappropriate sounds when their gate (sounding) is shortened. Cut a bell by four and it becomes a blip. By 20? Maybe you won't even hear a click.

The Folly of Software Delays: Ask the average BASIC programmer to force a half second delay and they'll throw a FOR NEXT loop at you. There is a certain amount of tweaking involved there to get the desired duration. This isn't the best way to do a delay. Anyone who owns a compiler knows that compiling a program muffs up delay loops and your sounds all sound differently. *TI* is a system variable, and it changes every 60th of a second. The natural way to delay for a half second is to count off 30 jiffies[1]:

```
100 x = ti 110: if ti-x <30
then 110
```

Need a quarter second? Test for less than 15. 3 seconds? <180.

If more games had lines like these instead of arbitrary FOR NEXT loops, they would run perfectly on all systems. The above code will wait for a half second, compiled or uncompiled, accelerated or unaccelerated, or any combination thereof -- with any type of accelerator.

In machine language people tend to increment .X and .Y thousands of times in a loop, tweaking the loop to generate enough of a delay. Unfortunately if you throw that code into an accelerator, it will run too fast. If you throw it into a slower running C-64 emulator, it will run too slowly -- and the delays will be increased accordingly.

In machine language, the thing to do is base your delays on either the hardware clock or the software clock, both of which run at the same speed regardless of the system speed. You could even count RDTIM ($FFDE)[2] and count the changes in the .Y register. Since your interrupt code is called every 60th second, you could set up dedicated timers that set off flags and reset themselves.

[1]Jiffy is a measure of time: 1/60th second.

[2].RDTIM is a kernal routine, usually used with machine language programs.

Compiler Timing

By Jeff Jones. I don't expect a huge reduction in FOR NEXT loops because of the previous five paragraphs. These are rules that Fender and I will have to live by since we'll want all of our software to work equally well on all systems.

When your perfectly timed sound is compiled or accelerated, you may run into timing problems. Unfortunately the average delay loop, `ford=1to1000:next,` isn't sped up much by normal compiling. Here's a comparison chart for the code:

```
10 a=ti
20 fori=1to2000:next
30 printti-a
```

Note that I had originally tested a program that counted to 1000, but realized that on a CMD accelerated system with Abacus compiler #2, less than one jiffy would accumulate. Since we need at least one jiffy for testing purposes, I doubled all the numbers. As you can see, the average BASIC delay loop can be accelerated by negligible amounts with normal compilers, but by a factor of four when Abacus' *all-integer* compiler #2 is used. This is because no floating point math is involved. We begin to see a simple way to keep delay loops of the same length whether compiled or uncompiled.

Given that it will always take a normal C-64 124 jiffies to count to 2000, we can use the loop as a test during setup and come up with a factor, *f*, with which we can equalize all subsequent delay loops.

For instance, if we're running a Blitzed program on a CMD 20MHZ accelerated system, the loop will probably take 96/20 or 4.8 jiffies. I'm dividing by 20 since the accelerated program will run 20 times faster. In our test loop, it will probably come out to four jiffies even. Since our standard is 124 jiffies for this loop, we take 124 and divide it by four and come up with a factor of 31. So all delay loops in that

The Underground

The 8-bit Commodore Mini-mag

The Underground is a bi-monthly publication with articles featuring reviews, PD/Shareware, BASIC and machine language tutorials, Geos, telecommunications, international stuff, news and tips. The Underground is a publication created by people who refuse to let their 8-bit computers gather dust, and it's priced low enough for even the most frugal users. Try a sample issue and see if The Underground is worth your while. You'll be glad you did.

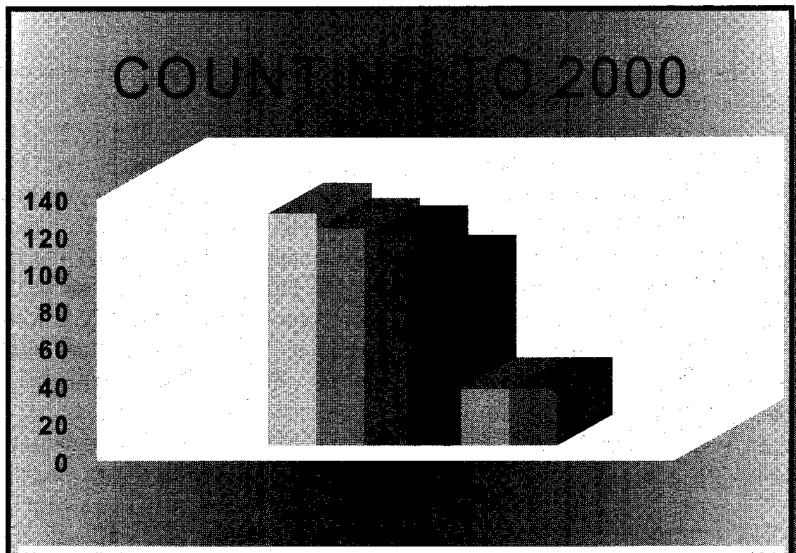
Single Issue: $2.00
Canada/Mexico$2.25
International $3.75

One year (six issues) $11.00
Canada/Mexico $12.50
International $21.50
Send check or money order (US funds) to:

The Underground
4574 Via Santa Maria
Santa Maria, CA 93455
USA

compiled and accelerated program should be multiplied by a factor of 31. Here is how the factor routine would look:

```
3000 f = ti
3010 fori=1to2000:next
```



Uncompiled	124
Abacus Compiler #1 P-code	116
Abacus Compiler #1 M-code	112
Blitz!	96
Abacus Compiler #2 P-code	30
Abacus Compiler #2 M-code	30

```
3020 f = ti - f
3030 if f = ti then f = 1 :
return
3040 f = 124 / f : return
```

Line 3030 exists because the jiffy clock might be off for some reason (you may have previously disabled the STOP key with POKE 788,52). In that case, you can't know how many jiffies have passed so your factor will have to be one. I suggest enabling the STOP key with POKE 788,49 during the test. Now that *f* is defined, we can insert it in all FOR NEXT type delay loops: `fori=1to500*f:next` will run at roughly the same speed, compiled or uncompiled, accelerated or unaccelerated, or any combination thereof. You can apply this scheme to the sounds supplied under this heading if you choose. I didn't include a factor in the routines since it would make all delays nil unless you first define *f*. That would make me a bit overbearing, wouldn't it? But we do ask that you use such a scheme if you submit a program with sound to LOADSTAR. We do want our programs to work satisfactorily on all systems.

Compiler Integers for Speed

By Jeff Jones. It may seem rather disappointing that compiling a FOR NEXT loop with *Abacus' Basic 64 Compiler* only saves you a couple of jiffies in most circumstances. There's a simple reason for this, and a way to increase your loops many fold.

The easiest way to make your compiled programs run faster is to use compiler #2, plain and simple. Compiler #2 is many times faster than compiler #1. Trouble is you can't always use compiler #2 since it deals only with integers between -32768 and 32767. So you may have to "mix "compilers using compiler directives in REM statements:

```
1000 REM@  01
```

switches to compiler #1 code

```
1070 REM@ 02
```

switches to compiler #2 code.

The reasons for the apparent slothfulness of compiler #1 are not numerous. It's all in the floating point math. A straight FOR NEXT delay and PRINT statements aren't sped up much at all by compilers. Your POKEs and computations and things going on *inside* loops will be sped up equally when comparing compilers #1 and #2.

If you can make the loop control variable, *LCV*, an integer variable, you'll greatly enhance the speed of your loops. Trouble is in regular BASIC a line like:

```
FOR I% = 1 TO 1000:NEXT
```

will crash the computer! It's simply not allowed to control a FOR loop with an integer LCV! The compiler allows it though. But if you want your program to run in BASIC as well as compiled, you can't have improper (to BASIC) syntax.

Using compiler #2 makes all variables integer whether you declare them that way or not. That means no variable can be greater than 32767 or less than -32768. If you must use a *real* (floating point) variable inside a compiler #2 program, you can declare variables as real at the start of the program with the following directive:

```
10 REM@R=1,J,K,L
```

This directive makes *I, J, K,* and *L* real variables, even in a compiler #2 program. Using these directives and strategies, you can create the best of both worlds within your compiled program.

The Old ROM Problem

By Jeff Jones. Parts of this article originally appeared on LOADSTAR #85, but I've learned a few things since then. Most important, I've learned how ML programmers can avoid common mistakes that will make their programs impossible to use for people with "old ROM" computers.

Commodore, in its infinite wisdom, has changed the C-64 over time. These changes have meant little to BASIC programmers, but much to ML programmers. The important change is simple: All C-64s DO NOT handle color memory the same way! I'm not going to get into which versions handle color which way. Frankly I can't remember. What's important are *which programming techniques we use to get around the problems.*

Many ML programmers mistakenly believe that if they clear the screen, color memory is filled with the color of the cursor. *We can't afford to be that lazy.* That may be true on your 128 in the 64 mode or any C-64 with JiffyDOS and most true C-64s, but there are some old C-64s that don't fill color memory at all when you clear the screen. There are some that fill color memory with the background color.

Result: You begin poking text and menus and other things to the screen yet *nothing* shows up. The character color is the same as background color. *Then the user calls up **LOADSTAR** and complains . . .* Oops! Back to programming:

What the old ROM user sees is nothing. Any time you POKE to the screen but don't manually set the color, there are many users who may not be able to see it.

The quick and dirty way to fix this is to set the cursor color to the desired fill color AND set the background to the fill color, then CLR the screen. You would have to poke the screen color back to the normal color when done, which would cause a flash unless you blank the screen.

Talk of such trickery is unsettling my stomach. This is *not* good programming practice! If you're going to poke to the screen, a better way to do it is through the screen and color pointers.

In BASIC screen-poking is usually a no-no because it's slower than a slug on a salt flat. PRINTing in BASIC is about as fast as CHROUT is in ML (at least to the human eye). However in ML, you'll find that POKEing is a heck of a lot faster than CHROUT because you're optimizing output.

```
LDA #147 ;clear character
JSR CHROUT
JSR  OLD'ROM  PATCH;  explained
later

LDY #119
POKE'BOX LDA #160
STA (209),Y
LDA desired color
STA (243),Y
DEY
BPL POKE'BOX
```

Longer code but faster code. And note that I used 160 to poke to the screen, since we're dealing with screen code, not ASCII.

But what's this (209) and (243)? We never set those pointers -- did we? Yes we did. (209) points to screen memory. Specifically, it points to the beginning of the line the cursor is on (not to the cursor itself.) If the cursor is on the 8th row, the cursor would be at (209),Y with Y being 8. (209) always points to the left HOME of each line.

So when you clear the screen, 209, the low byte of the pointer is zero. 210, the high byte is four.

4 * 256 + 0 = 1024

Move the cursor down one line? The operating system automatically updates 209 and 210.

209 = 40 210 = 4
4 * 256 + 40 = 1064

That's where row one starts --1064.

Here's where things get hairy! In some old ROM computers the color pointer IS NOT updated by the operating system. Not until a character is printed with CHROUT. When you think about it, it's more efficient, but hackers with late model 64s have no idea that some computers haven't updated this pointer. So if you use the 243 pointer to begin poking to the corresponding color memory, it may work well on YOUR computer, but some poor guy in Texas with a ten-year-old C-64 watches his computer crash. This is because 243 could be pointing anywhere -- ANYWHERE in memory. Get it? You're poking color right into your program or zero page or your font . . . All because the color pointer hasn't been updated when you moved to the current line.

The way to get around this is to JSR to OLD ROM PATCH whenever you change lines. This is ONLY if you're poking to color memory through the 243 color pointer. If you're printing with CHROUT, you don't need OLD ROM PATCH.

```
OLD'ROM'PATCH LDA 210
SEC
SBC 648
CLC
ADC #$D8
STA 244
LDA #209
STA 243
RTS
```

Okay, okay, don't go grumbling about this taking up too many cycles. This is machine language here! This subroutine is called and returned in approximately 3/100000th sec including the RTS. I don't think it'll make your program crawl!

If you don't call the OLD ROM
PATCH after moving between lines, your color pointer may NOT be pointing to the right place. This can cause crashes and theoretically tendonitis and higher taxes.

My famous box routine uses the
OLD ROM PATCH, which can be called up to 25 times, depending on the number
of lines in your box. Remember, that's only 3/4000th sec added to the drawing time.

Jeff's ML Box Routine

```
      LDX Y1
      LDY #0
      CLC
      JSR PLOT
      JSR OLD'ROM'PATCH
      INC Y2
      INC X2

 ....SET'UP LDY X1
 . . .DO'BOX LDA CODE
 . . STA (209),Y;  POKE REVERSE
SPACE
 . . LDA COLOR
 . . STA (243),Y;  POKE COLOR
 . . INY
 . . CPY X2
 . . .BCC DO'BOX
 .    LDA 214;   WHAT LINE AM I
ON?
 .    CMP X2
 . . .BEQ QUIT
 . . LDA #13
 . .  JSR CHROUT; MOVE DOWN A
LINE
 . . JSR OLD'ROM'PATCH
 ....JMP SET'UP
   ..QUIT RTS

      X1 .BYT 0
      X2 .BYT 0
      Y1 .BYT 0
      Y2 .BYT 0
      CODE .BYT 0
      COLOR .BYT 0
```

This routine will draw a box anywhere on the screen. Note how I used CHROUT to move to the next line, automatically updating (209). This method is very fast. Faster than printing. Printing carriage returns DOES slow down the process by a few thousandths of a second. So if you're in a real hurry to fill the screen, just add #40 to 209 and 210 instead of printing a carriage return. You could also build the OLD ROM PATCH right into the routine so you don't lose cycle time on the JSR and RTS. Seriously though, you shouldn't have to worry about this much efficiency except in raster interrupt work. And this type of code wouldn't be in the raster interrupt routines anyway.

A Machine Language Call To Arms

Machine language programmer, you're a special breed -- even a dying breed. As computers become faster and faster, there will be little need for anyone but operating system authors and compiler authors to know machine language.

If you can learn 6510 assembler then you have a mind meticulous enough to handle a $450 C compiler for your Pentium and secure a future. Don't feel daunted by the thick manuals, and don't let anyone tell you that programming a C-64 is easier than programming an IBM. We have fewer tools right down to the processor level. Frankly, while the C-64 is a very accessible machine, it's a bit harder to program than the IBM. I have whole libraries of standard routines at my disposal when I program the PC. I had to write my own low level routines for the C-64.

While our policy does call for BASIC programs, we respect ML programs. We just hate ML crashes. *Please Don't Be Tricky!* Don't try to give us the smallest, smartest fastest routine. This is a lesson learned well in the PC world. Size doesn't matter. It's the end product *delivering* that counts. We just want it to work for everyone and be noticeably fast. If it must be slower for maximum compatibility, let it be slower. If it must be bigger, let it be bigger.

We also want to be able to edit your ML programs. This may only be to make color or font changes. One way to make editing easier for us is to give us a JMP table that allows us to change the initial setup. You can make this self explanatory by having a boot file that pokes or SYSes parameters to preferences within your program. REM each line so that we know how to change each parameter. That way we don't have to hack your ML at all.

The number one way that I hack ML is to find all the instances of file access that poke hard coded 8s into the current drive before calling the ML. Your ML shouldn't LDX #8 when it uses SETLFS. It should LDX DEVICE which is a variable somewhere in your program.

The more variables like DEVICE that you create, the more easily configurable your program becomes. You may come up with parameters that no one will ever change, but at least the ones that YOU want to tweak become easy. And if you want to make it user-configurable, you'd have to write it that way anyway.

If you change your mind about a color or position choice, you only have to change the parameters in one place instead of hunting down each instance where you prepped the cursor to print a certain item. When you move to a parameter-driven program, it feels as if you're wasting memory, but you soon realize that while your source code is growing, your object code suffers little. Each new setting only takes up one byte. You could conceivably have a definable setting for every item on your screen and not take up more than 100 bytes of variable space. But what you're left with is a program that we can tweak until it's at its prettiest --without wading through a disassembly listing.

RAM Under ROM

By Jeff Jones. Here's how you can access the RAM under all the ROM, even the IO at $D000-$DFFF. I use these subroutines to knock all ROM images out so that I can get a byte from beneath them, and then on again so that everything continues as normal. Generally I call ROM'OUT right before I need to LDA the data, and call ROM'IN immediately after so that any interrupt driven software suffers the least. These routines don't affect any register.

```
ROM'OUT PHA
LDA #$74
SEI
STA 1
PLA
RTS
```

```
ROM'IN PHA
LDA #$77
STA 1
CLI
PLA
RTS
```

When speed is important, I'll go ahead and write the ROM out/in routines into the larger routine that needs it. I might do whole chunks of work with the ROM out and no interrupts (which is faster), but usually ML is fast enough that I can have a zillion JSRs to ROM'IN and ROM'OUT without anyone noticing.

Remember that ROM must be OUT if you want to write to the RAM between $D000 and $DFFF. The ROM can be IN when writing to the RAM under the BASIC and KERNAL ROMs, but has to be OUT only when reading from them.

Symbolic Assemblers

With assemblers, you can fine tune your code until it shines. I recommend ROCK'S ASSEMBLER, available on the *Compleat Programmer.* See our ad on page 7. I know I don't use it personally, but that's not because I don't like it. It has quite a few features that I wish existed in EBUD. I personally don't use ROCK'S ASSEMBLER because I have become too accustomed to EBUD. I'm just a creature of habit.

Using most any other assembler can cause problems. For instance, MERLIN is widely used, but it's so archaic, cryptic and limited compared to these two assemblers that I can't recommend it. If you're reading this and produce or sell a assembler that you feel is better than EBUD or ROCK'S ASSEMBLER, send it to me and I'll review it. Note that we don't do negative reviews so don't worry about me trashing your product.

There I go digressing again! No one in this age should write code in a monitor or poke ML into memory from BASIC. If you're going to program ML, do it big and do it right. ML Monitors that allow you to assemble code here and there used to be called assemblers. Now they are considered less than assemblers because they do little to help you edit. They are handy for inspecting and writing little patches, but when you start writing big programs with them, it becomes too much of a problem to fix mistakes. You end up patching in spaghetti-like routines to fix or upgrade, and slowly you forget what's going on in your own code.

Without a symbolic assembler, I'd probably STILL be writing our presenter. There are roughly 70 blocks of text in LOADSTAR BRIEFS. The assembler allowed me to type some, edit it, correct mistakes, assemble whole files of text into the program, and print it.

Machine language programs are no longer merely tiny support modules that serve a larger BASIC program. Large machine language programs are rather commonplace. And these programs are chock full of help and features. I say that if you're going to enhance your BASIC program with ML, or write a 100% ML program, buff it up with color and text. You have more memory available with ML. Use it.

LOADSTAR® MONTHLY

LOADSTAR is a monthly "magazine on disk" for the Commodore 64/128. Subscribers receive two 1541 disks (or one 1581 disk) in their mailbox every month packed with news, articles and programs. These non-PD, high-quality programs are written by the best home-based programmers in the field and edited by the crack LOADSTAR team of Fender Tucker and Jeff Jones. Subscription prices are at an all-time low of $69.95 for a 12-month subscription, or $19.95 for a three-month subscription. You may also elect to subscribe "by the month," where we charge your credit card $7.95 for each issue after it's shipped. We also offer the long line of standalone products below.

✔ **NEW Games Disk! The Compleat Jon:** 11 Games! The whole gamut of gaming is covered here: artificial intelligence, role-playing, mazes, fantasy, science fiction, education and even non-violence (which was a radical concept in its time). These eleven games are among the best ever published on LOADSTAR. Listed on the menu in chronological order, so you can see how Jon's style changed as the years rolled by. **1581 disk 0021D3 $20. 1541 disk #0038D5 $20**

✔ **NEW Puzzle disk! The Compleat Crossword**: Every crossword puzzle published in Puzzle Page in one huge collection! 220 puzzles! It uses Barbara Schulak's CRUCIVERBALIST program to present the puzzles and allows you to "mark" a puzzle when it's solved so that you know which you've solved and which you haven't yet. Each 1541 disk contains 110 puzzles. **1581 Disk #0020D3 $20. Disk 1 (1541) #0036D5 $10.Disk 2 (1541) #0037D5 $10**

✔ **NEW C-128 Productivity! The Compleat Lee O:** Six of Lee O. Clinton's best serious programs for the C-128 80- column mode. Finance, auto expense, kitchen helper, genealogy, resume writing, mutual funds! **One 1541 disk #0032D5. One 1581 disk #0017D3 $10.00**

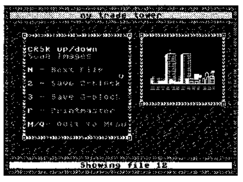
✔ **NEW Word Search! Super Star Search 1:** 200 original word search puzzles by Steven Thomas and Art Dudley, presented by John Serafino's modern point and click program. **One 1541 disk #0011D5 One 1581 disk #0008D3 $20.00**

✔ **NEW Story disk! The Compleat Prosequest '95:** NEW!!! A 1541 disk with all of the entries in the 1995 short story writing contest on it, including the three grand winners. **One 1541 disk #0035D5 One 1581 disk #0019D3 $5.00**

Soft Wear! LOADSTAR T-Shirts: Limited edition Fruit Of The Loom T-shirts. LOADSTAR's nemesis, Knees Calhoon, stands up to regular washing and drying. Where else can you find Commodore apparel? 50% Cotton//Polyester. **$15.00 each with free shipping! Small #960025, Medium #960125, Large #960225, X-Large #960325, XX-L #960425**

Tools! Tutorials! The Compleat Programmer: Best seller! Over Two megabytes of knowledge crammed and stuffed onto eight 5.25-inch disks or two 1581 disks! Plus tools, extensions, languages, assemblers, tutorials and utilities! **5.25 set #0005D5. 3.5-inch disk #0005D3. For $5 more, get C= Hacking MAG #0006D3 (on 3.5-inch disks only and NOT available separately)** to complete your programming set. **$20.00**

PS Graphics! Compleat PS Vol. 1 (The Print Shop by Broderbund or Printmaster required). Over 1300 artistic and never before published PRINT SHOP images. Scan through the many PRINT SHOP images sequentially, by name, or by group number. Press a key and save the graphic you want in 2-block, 3-block and even PRINTMASTER graphic files! All that plus a printed guide! Each volume is $20.00. **Vol. 1: C-64/128 3.5-inch disk item #0001d3. 5.25-inch disks item #0009d5. Vol. 2 (graphics from past LS issues): C-64/128 3.5-inch disk item #0002d3. 5.25-inch disks item #0010d5.**



GEOS Clipart

LOADSTAR presents the biggest Geos collection of clip art and fonts ever offered at one time. All of the Geos art that's ever appeared on LOADSTAR, as well as some great files from Geos fanatic Dick Estel, are available on twenty 5.25 inch disks or eight 3.5 inch disks. Most of this has never been seen before! Use these graphics in your GeoPaint, GeoWrite and GeoPublish documents or convert to FGM with FGM utilities. Spiff up your GeoFAX documents with the appropriate graphic -- every time! Prices are $20 for any two 3.5 inch disks, or any five 5.25 inch disks. You can purchase the whole collection for $75 for either version. Call LOADSTAR toll-free at 1-800-594-3370 or 1-318-221-8718 to order by credit card. Or send check or money order and specify (by LG number) which disks you want.

5.25-INCH DISKS

Disk 01 - RAILS: Railroad art from Europe and the USA **#0012D5**

Disk 02 - VEHICLES/TAROT: Artwork of old and new autos; excellent geoPaint drawings of the Tarot card set **#0013D5**

Disk 03 - CLIP ART: Includes converted MacPaint files that have never before been available in Commodore format **#0014D5**

Disk 04 - OTTAWA/PRIME CLIPS: Artwork of the main landmarks of Ottawa; plus high quality public domain clip art **#0015D5**

Disk 05 - FONTS: More than 30 fonts from past issues of LOADSTAR, plus articles (in geoWrite format) on creating fonts. Also two ready-made headers for use with your own documents, one a picture of a mail truck; the other reading FROM THE DESK OF **#0016D5**

Disk 06, Disk 07, and **Disk 08** - geoPaint and Photo Album files with the great clip artwork featured on past LOADSTARS - Includes GeoCurmudgeon, Anamalia I and II, Australian Animals, Valentine art and many more **#0017D5, #0018D5, #0019D5**

Disk 09 - GOODYKOONTZ FILES - Jasper Goodykoontz, born in Indiana in 1855, produced Goodykoontz's Perpetual Calendar and General Reference Manual (A Book for the Millions). This disk includes scans from the book of a wide array of subjects -- Gestures and Attitudes, Poultry, Craniology, and more. **#0020D5**

Disk 10 - OLD WEST: Scanned Artwork from Dick Estel's FRD Software - mostly woodcut style art of the old west, gold rush days and pioneer scenes. **#0021D5**

Free Shipping! For Every item on this page!

Disk 11: J. Neely Art/Animals: Jennifer Neely works with a wide variety of subject matter and materials. Disk contains some of her favorites, scanned into geoPaint format. Side 2 is a collection of scanned artwork of animals from FRD Software **#0022D5**

Disk 12 - HOLIDAY: Artwork for New Years, Valentine's, St. Patrick's Day, Halloween, Thanksgiving and Christmas **#0023D5**

Disk 13 - PEOPLE/FACES: Scenes of people and faces from FRD Software **#0024D5**

Disk 14 - FRD CLASSICS: Dick's choice of the best of the FRD collection **#0025D5**

Disk 15 - DINOS/CLASSICS: Dinosaurs and other prehistoric beasts, as well as more first choice artwork from FRD. **#0026D5**

Disk 16 - SPORTS/MISC: Dozens of sports-related clips **#0027D5**

Disk 17 - OFFICE AND SCHOOL: Clips to be used at work and around the house **#0028D5**

Disk 18 -MUSIC & MORE SCHOOL CLIPS #0029D5

Disk 19 - SEASONAL AND HOLIDAYS: A clip for any occasion **#0030D5**

Disk 20 - SEASONAL AND HOLIDAYS: A clip for any occasion **#0031D5**

3.5 INCH DISKS

The 3.5" disks are roughly equivalent to two and a half 5.25" disks.

Disk 1: Equals disks 1, 2, 4B #0009D3
Disk 2: Equals disks 3, 6, 7A #0010D3
Disk 3: Equals disks 5, 8, 7B #0011D3
Disk 4: Equals 9, 10, 11A #0012D3
Disk 5: Equals 12, 13, 11B #0013D3
Disk 6: Equivalent of Disks 14, 15and some bonus files ~~not on 5.25" disks~~ **#0014D3**
Disk 7: Sports, Office and school, Music #015D3
Disk 8: Music, Holiday and Seasonal #016D3

For your convenience, GeoViewer is included on each volume. GEOS 2.0 is suggested.

Diskfulla Card Games! The Compleat Maurice: A compilation of 26 solitaire card games written by Maurice Jones, the acknowledged master of card game simulations for the C-64/128. There's even a brand new, never before published game called Boomerang. **Two 5.25 inch disks #0007D5 or one 3.5 inch disk #0007D3. $20.00 postage paid!**

Oodles Of Stunning Art! Tutorials! The Compleat Walt: During LOADSTAR's first ten years we have published 24 of Walt Harned's slideshows and multimedia events. Now we've gathered them into one huge collection: seven 5.25 inch disks or three 3.5 inch disks! There are over 250 pictures, including some that have never been published. The greatest one-man collection of art on any computer platform. **5.25-inch disks order #070425 3.5-inch disks order #070423. $20.00 postage paid!**

Loadstar Order form ☎ 1-800-594-3370

QTY	Description	Item #	Price ea.	Total

Use extra sheet of paper for large orders

Name ______________________________

Address ______________________________

City ______________ **State** ______ **Postal Code** ______

Total Enclosed:______________

❑ Check/money order made payable to "Loadstar" in US Funds
❑ MasterCard ❑ Visa ❑ American Express ❑ Discover
Card #______________________________
Exp Date _____/_____ Authorized signature: ______________
Subscribers: I prefer ❑ 5.25-inch diskettes ❑ 3.5-inch diskettes

LOADSTAR
P.O. Box 30008, Shreveport, LA 71130-0008
Questions: 1-318-221-8718 Fax 1-318-221-8870

CMD • HARDWARE • ORDERS: 1-800-638-3263 • SOFTWARE • CMD

HD Series SCSI Hard Disk Drives

HD-40, 40 MB (Limited Supply)	**CALL**
HD-170, 170 MB (Special Edition)	**$319.00**
HD-500, 500+ MB (Special Edition)	$399.00
HD-1000, 1 GB (Special Edition)	$499.00
HD-2000, 2 GB (Special Edition)	$899.00

FD Series 3.5" Floppy Disk Drives

FD-2000 (800K and 1.6 MB)	**$179.95**
FD Real-Time-Clock Option/Kit	$20.00/$29.00
Box of 10, High Density Disks (1.6MB)	$14.95
Box of 10, Enhanced Density Disks (3.2 MB)	$29.00

JiffyDOS

(Specify computer serial number and drive model)

JiffyDOS C64/SX-64 System	$49.95
JiffyDOS C-128/128-D System	$59.95
Additional Drive ROM's	$24.95

Miscellaneous Hardware

80-column Monitors (Refurb)	CALL
Action-Replay Cartridge	**$59.95**
Aprotek 3-Way User Port Expander	$30.00
Aprotek User Port Extension Cable	$19.00
C-64 Computer (Refurb, w/JD)	**$99.00**
C-64C Computer (Refurb, w/JD)	**$119.00**
C-64, C-64C Power Supply (Repairable)	$39.00
C-64, C-64C HD Power Supply (Repairable)	**$56.95**
C-128 Computer (Refurb, w/JD)	**$179.00**
C-128 Computer (Refurb, w/JD, no PS)	**$129.00**
C-128 Power Supply (Repairable)	**$56.95**
Cannon BJC-4100 Color Bubble Jet Printer	$399.00
Commodore 1541 Disk Drives (Refurb, w/JD)	**$89.00**
Commodore 1541-II Disk Drive (Refurb, w/JD)	**$109.00**
Commodore 1571 Disk Drive (Refurb, w/ JD)	**$119.00**
Commodore 1702 40-column Monitor (Refurb)	**$109.00**
Commodore 1802 40-column Monitor (Refurb)	**$139.00**
Commodore SX-64 Computer (Refurb, w/JD)	**$199.00**
CMD EX2+1 3-Port Cartridge Port Expander	$34.95
CMD EX3 3-Port Cartridge Port Expander	$29.95
CMD Gamepad/Joystick Controller	$24.95
CMD GeoCable II Print Cable 6ft./15ft.	$29.00/$34.00
CMD SmartMouse (1351 Compatible Mouse)	$49.95
CMD SmartTrack (1351 Compatible Trackball)	$69.95
Inkwell Light Pen Model 170C	$75.00
Monitor Cables	CALL
Mouse Pad	$2.95
MW-350 Printer Interface (ØK/8K Buffer)	$49.00/$60.00
Panasonic KXP-1150 9-pin Printer	$169.00
Panasonic KXP-2023 24-pin Printer	$229.00
Panasonic KXP-2135 24-pin Color Printer	$319.00

Utilities

Big Blue Reader V4.10 (SOGWAP)	$39.00
CMD Utilities	$24.95
JiffyMON-64 (ML Monitor)	$19.95
The Compression Kit '94 (Mad Man)	$39.00

GEOS

Collette Utilities (Handy Geos Utilities)	$19.95
Desk Pack Plus	$29.00
Dweezils Greatest Hits (NewTools2,Stamp,Label128)	$30.00
FONTPACK Plus	$25.00
gateWay 64 or 128 (Specify Version)	$29.95
geoBASIC	$20.00
geoCalc 64/128	$40.00/$45.00
geoChart	$29.00
geoFAX	$39.95
geoFile 64/128	$40.00/$45.00
geoMakeBoot (Makes Bootable copies)	$12.95
geoProgrammer	$45.00
geoPublish	$40.00
GEOS 64 v2.0	$44.00
GEOS 128 v2.0	$49.00
geoSHELL V2.2 (CLI for GEOS)	$24.95
International FONTPACK	$25.00
Perfect Print LQ for GEOS (Laser-like output)	$49.95
RUN GEOS Companion	$20.00
RUN GEOS Power Pak I or II (Specify)	$20.00

Books

Anatomy of the 1541	$12.00
C-64 Science & Engineering	$12.00
C128 Computer Aided Design	$12.00
C128 BASIC Training Guide	$12.00
Commodore 64 Tricks and Tips	$12.00
GEOS Programmers Reference Guide	$35.00
Graphics Book for the C-64	$12.00
Hitchhikers Guide to GEOS	$35.00
Ideas for Use on Your C-64	$12.00
Printer Book for the C64	$12.00
Mapping the C64	$12.00
Simple Internet	$16.95

Scanning & Video

Handyscanner	$249.00
Pagefox	$139.00
Video Digitizer	$189.00
Videofox	$89.00

RAMLink Power-Backed RAM Disk

RAMLink Base Model (ØMB, No RAMCard)	$149.00
RAMLink w/1 MB RAMCard (Special!)	**$199.00**
RAMLink w/4 MB RAMCard (Special!)	**$299.00**
RAMCard RTC Option/Kit	$10.00/$19.00
RAMLink Battery Back-up (Optional)	$24.95
Parallel Cable (RAMLink to HD)	$14.95

MIDI & Sound

Digimaster	$34.95
SID Symphony Stereo Cartridge	$44.95
Sonus 64 Sequencer, MIDI Interface, Cables	$99.00
Sonus 128 Sequencer, MIDI Interface, Cables	$99.00

Languages & Compilers

BASIC 64 Compiler (Abacus)	$17.00
BASIC 128 Compiler (Abacus)	$25.00
Blitz! 64 Compiler (Skyles)	$30.00
Blitz! 128 Compiler (Skyles)	$30.00
Buddy 64/128 Assembler	$39.00
Cobol 64 (Abacus)	$17.00
Fortran 64 (Abacus)	$17.00
Pascal 64 (Abacus)	$17.00
Power C 64 (Spinnaker)	$14.00

Productivity

Cadpak 64 (Abacus)	$22.00
Cadpak 128 (Abacus)	$25.00
Chartpak 64 (Abacus)	$17.00
Chartpak 128 (Abacus)	$25.00
Data Manager 64 (Timeworks)	$16.00
I Paint v1.5 (128, 80-col., 64K VDC) (Living Proof)	$39.00
I Port v1.54 (128, 80-col., 64K VDC) (Living Proof)	$29.00
Personal Portfolio Manager (Abacus)	$16.00
Pocket Writer 1 (64) Digital Sol.)	$35.00
Pocket Writer 2 (64/128) (Digital Sol.)	$65.00
Pocket Writer 3 (64 or 128) (Digital Sol.)	$70.00
Pocket Planner 2 or Pocket Filer 2 (Digital Sol.)	$35.00
RUN Productivity Pak I, II, or III (Specify)	$15.00
RUN Super Starter Pak 1541 or 1581	$20.00
RUN Works	$20.00
SEC Check Register 128	$29.00
SuperScript 64 (Precision)	$15.00
SuperScript 128 (Precision)	$20.00
Suberbase 64 Version 3.01 (Precision)	$35.00
Superbase 128 Version 3.01 (Precision)	$35.00
SwiftCalc 64 (Timeworks)	$16.00
TWS 64 w/Speller (Busy Bee)	$29.00
TWS 128 w/Speller (Busy Bee)	$39.00
TWS Modules (HD/RL/Illustrator)	each $5.00

Telecommunications

Aprotek MiniModem C-24 (C= ready, 2400 baud)	**$69.00**
Aprotek MiniModem C (C= ready, 1200 baud)	**$50.00**
Aprotek Com-Modem Adapter (for ext. modem)	$19.00
Aprotek Com-Modem Adapter Cable	**$9.95**
BOCA 2400 Baud Modem	$69.00
BOCA 2400 w/SwiftLink and Cable	$99.00
BOCA 14.4K bps FaxModem	$129.00
BOCA 14.4K w/SwiftLink & Cable	$159.00
BOCA V.34 28.8K bps FaxModem	$199.00
BOCA V.34 w/SwiftLink & Cable	$229.00
Commodore 1670 Modem (1200 baud)	**$14.95**
Dialogue 128	$29.00
SwiftLink RS-232 Cartridge (Up to 38.4K baud)	$39.95
SwiftLink Modem Cable (DB9-DB25)	$9.95

Games

Atomino	$17.00
Ballistix	$13.00
Blood Money	$15.00
Chomp!	$10.00
Day in the Life of Prehistoric Man	$19.00
Escape Route	$19.00
Heavenbound	$19.95
Island of the Dragon	$19.00
Laser Squad	$13.00
Lazer Duel	$14.95
Lions of the Universe	$19.95
Mainframe	$13.00
Menace	$15.00
Navy Seal	$10.00
Rings of Medusa	$16.00
RUN C128 Funpak	$10.00
RUN C64 Gamepak	$10.00
Skate or Die	$13.00
Slaterman	$14.95
The Amazing Spider-Man	$15.00
The President Is Missing!	$10.00
The Three Stooges	$10.00
Tie Break Tennis	$16.00
Total Eclipse	$10.00
Ultima V	$17.00
Walkerz	$19.00
Wings of Circe	$19.00
Wizardry 5: Heart of the Maelstrom	$24.00

CMD Repairs Computers & Drives!

WE ACCEPT ALL MAJOR CREDIT CARDS

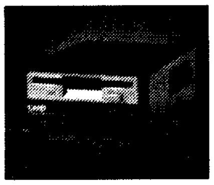
JiffyDOS

Increase Speed Up to 1500% while retaining 100% compatibility

- Speeds up Loading, Saving, Verifying, Formatting and Reading/ Writing of Program, Sequential, User and Relative files unlike cartridges which only speed up Loading and Saving of PRG files
- Built-in DOS Wedge plus 17 additional features including file copier, text dump, printer toggle, and redefinable function keys make using your computer easier and more convenient
- ROM upgrade installs easily into most computers and disk drives. Supports C-64, 64C, SX-64, C-128, 128-D, 1541, 1541C, 1541-II,1571, 1581 and more.

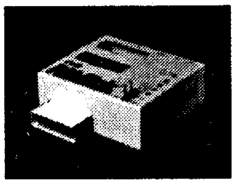
FD Series

1.6 MB 3.5" Floppy Drives



The FD-2000 disk drives utilize today's latest 3.5 inch disk drive technology. FD-2000's support 800K (1581 style) and 1.6 MB (High Density) formats. Fast and reliable, they support 1541, 1571 and 1581 style partitions, Native Mode partitioning and can actually read and write 1581 disks. FD's feature built-in JiffyDOS, SWAP button and optional Real-Time Clock. High capacity, speed and compatibility make the FD right for every application, including GEOS.

RAMLink

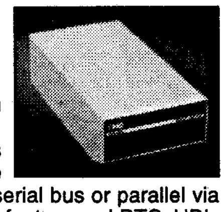


Power-Backed Expandable REU

The fastest possible form of storage, RAMLink provides instant access to files and retains data while your computer is turned off. Easy to use and expandable up to 16 MB, RAMLink utilizes the same powerful operating system found in CMD Hard Drives. Unlike Commodore REU's which are compatible with less than 1% of commercial software, RAMLink supports more than 80% of the commercial titles. RAMLink also offers built-in JiffyDOS Kernal, SWAP feature, reset button, enable/ disable switch, pass-thru port and RAM port for REU/GEORAM use. RAMLink offers maximum speed, expandability and compatibility with all types of software and hardware including GEOS.

HD Series

SCSI Hard Drive for the C-64/128



HD Series Hard Drives are available in capacities up to 1 GB, are fully partitionable, and can emulate 1541, 1571, & 1581 disks while Native partitions utilize MS-DOS-style subdirectories. HD's connect easily to the serial bus or parallel via RAMLink. Includes built-in JiffyDOS, SWAP feature and RTC. HD's offer superior compatibility with most commercial software including BBS, Productivity and GEOS. And with new pricing, HD Series drives offer the lowest cost per megabyte of any C64/128 storage device.

Shipping and Handling Charges

Use the chart below to match your order subtotal with your shipping zone and method

	Continental United States					
Order Subtotal	UPS Ground	2nd Day	Next Day	AK HI, PR	Canada	Foreign
$0.01 to $19.99	$4.50	$13.00	CALL	$15.00	$5.00	CALL
$20.00 to $29.99	$5.50	$17.00	CALL	$17.00	$7.00	CALL
$30.00 to $59.99	$6.50	$19.00	CALL	$19.00	$9.00	CALL
$60.00 to $149.99	$7.50	$21.00	CALL	$28.00	$12.00	CALL
$150.00 to $299.99	$10.00	$25.00	CALL	$35.00	$20.00	CALL
$300.00 to $799.99	$15.00	$29.00	CALL	$40.00	$25.00	CALL
$800.00 and up	$20.00	CALL	CALL	$50.00	$38.50	CALL

UPS C.O.D. add $5.00 (US/PR only)

Policies: CMD accepts: Major Credit Cards, Money Orders, COD and Personal Checks. Personal Checks are held 3 weeks. Customer is responsible for shipping charges on refusals. Most items are stock, contact CMD for firm delivery. Returns for merchandise credit only within 30 days with prior authorization. Credits are less shipping, handling, taxes, customs, duties, and a 10% restocking charge. No refunds or credits on opened software. All prices and specifications are subject to change without notice.

Creative Micro Designs, Inc.
P.O. Box 646
East Longmeadow, MA 01028

Info: (413) 525-0023
Fax: (413) 525-0147